

ANALYSIS AND DETECTION OF INTRUSION DETECTION ON NETWORK SECURITY ENVIRONMENT USING ML

GAUTHAM R¹, NADIM SURAJ M², BHARATH M³, SAMUNDEESWARI M⁴

¹UG Scholar, Department of CSE, Kingston Engineering College, Vellore-59

²UG Scholar, Department of CSE, Kingston Engineering College, Vellore-59

³UG Scholar, Department of CSE, Kingston Engineering College, Vellore-59

⁴Asst.Professor, Department of CSE, Kingston Engineering College, Vellore-59

Abstract - The Intrusion Detection System (IDS) is a critical cyber security instrument that monitors and detects intrusion threats. This research intends to examine contemporary IDS research utilizing a Machine Learning (ML) methodology, with a focus on datasets, ML algorithms, and metrics. The choice of datasets is critical for ensuring that the model is suitable for IDS use. Traditional methods, such as firewalls, which focus on data filtering, may not be adequate for detecting all forms of assaults in a timely manner. Machine learning algorithms-based on intrusion detection systems (IDS) are especially good in efficiently processing vast amounts of data in order to identify any malicious behavior for effective handling and prompt detection of these types of attacks. Machine learning-based intrusion detection systems (IDS) are used to monitor all network traffic for malicious activity. In order to improve the intrusion detection system's detection rate, the system's focus was on false negative and false positive performance measures. The results of this paper showed that the ML model XGBoost classifier had the best accuracy rate, while the Decision tree classifier had the lowest.

Key Words: IDS, XGBoost Algorithm, Decision tree

1.INTRODUCTION

There are two types of IDS in general (anomaly base or misuse base). Anomaly intrusion detection system was designed to detect attacks based on normal activity that had been recorded. This form of intrusion detection system is commonly utilized because it can detect new types of intrusions by comparing current real-time traffic with previously recorded normal real-time traffic. However, it registers the highest levels of false positive alarm, implying that a huge number of normal packets are mistaken for attacks packets. A misuse intrusion detection system, on the other hand, is used to detect attacks using a repository of attack signatures. It does not generate false alarms, but it can be bypassed by a new type of attack (new signature).

Filtration of attack packets is required to prevent network attacks. Dynamic allocation should be used to put idle resources to work for cloud users and ensure that they receive high-quality service. The rivalry for resources is the main difficulty with intrusion detection. The conflict will be won by whichever side (attacker or user) has more resources. The conflict will be won by the side (attacker or user) with the most resources. The challenge with intrusion detection is one of resource management.

Intrusion can take several forms, such as flooding the network with a huge number of packets, synchronizing those packets, or creating zombies to attack the victim machine. Flooding the network with assault packets is the most basic and effective intrusion approach. Individual cloud clients have always been vulnerable to intrusion detection since they have fewer resources to combat such attempts. However, numerous cloud computing risks can be used to mitigate cloud threats.

A Denial-of-Service attack renders cloud resources inaccessible or depletes them for cloud users. The most typical strategy employed in such attacks is to flood the network of the targeted system with false packet requests, obstructing legitimate network traffic. The server is generally overloaded as a result of such attacks. Thousands of attackers target a single machine in a distributed attack.

2.RELATED WORK

[1] This work is done by author L. Liu, P. Wang, J. Lin and L. Liu, "Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning," in *IEEE Access*, vol. 9, pp. 7550-7563, 2021

In imbalanced network traffic, malicious cyber-attacks can often hide in large amounts of normal data. It exhibits a high degree of stealth and obfuscation in cyberspace, making it difficult for Network Intrusion Detection System(NIDS) to ensure the accuracy and timeliness of detection. This paper researches machine learning and deep learning for intrusion detection in imbalanced network traffic. It proposes a novel Difficult

Set Sampling Technique(DSSTE) algorithm to tackle the class imbalance problem. First, use the Edited Nearest Neighbor(ENN) algorithm to divide the imbalanced training set into the difficult set and the easy set. Next, use the KMeans algorithm to compress the majority samples in the difficult set to reduce the majority. Zoom in and out the minority samples' continuous attributes in the difficult set synthesize new samples to increase the minority number. Finally, the easy set, the compressed set of majority in the difficult, and the minority in the difficult set are combined with its augmentation samples to make up a new training set. The algorithm reduces the imbalance of the original training set and provides targeted data augment for the minority class that needs to learn. It enables the classifier to learn the differences in the training stage better and improve classification performance. To verify the proposed method, we conduct experiments on the classic intrusion dataset NSL-KDD and the newer and comprehensive intrusion dataset CSE-CIC-IDS2018. We use classical classification models: random forest(RF), Support Vector Machine(SVM), XGBoost, Long and Short-term Memory(LSTM), AlexNet, Mini-VGGNet. We compare the other 24 methods; the experimental results demonstrate that our proposed DSSTE algorithm outperforms the other methods.

[2] This Research is done by author M. Almseidin, M. Alzubi, S. Kovacs and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, 2017

Intrusion detection system (IDS) is one of the implemented solutions against harmful attacks. Furthermore, attackers always keep changing their tools and techniques. However, implementing an accepted IDS system is also a challenging task. In this paper, several experiments have been performed and evaluated to assess various machine learning classifiers based on KDD intrusion dataset. It succeeded to compute several performance metrics in order to evaluate the selected classifiers. The focus was on false negative and false positive performance metrics in order to enhance the detection rate of the intrusion detection system. The implemented experiments demonstrated that the decision table classifier achieved the lowest value of false negative while the random forest classifier has achieved the highest average accuracy rate.

[3] This work is done by another author X. Ma and W. Shi, "AESMOTE: Adversarial Reinforcement Learning with SMOTE for Anomaly Detection," in *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 943-956, 1 April-June 2021, doi: 10.1109/TNSE.2020

Intrusion Detection Systems (IDSs) play a vital role in securing today's Data-Centric Networks. In a dynamic environment such as the Internet of Things (IoT), which is vulnerable to various types of attacks, fast and robust solutions are in demand to handle fast-changing threats and thus the ever-increasing difficulty of detection. In this paper, we present a novel framework for the detection of anomalies, which, in particular, supports intrusion detection. The anomaly-detection framework we propose combines reinforcement learning with class-imbalance techniques. Our goal is not only to exploit the auto-learning ability of the reinforcement-learning loop but also to address the dataset imbalance problem, which is pervasive in existing learning-based solutions. We introduce an adapted SMOTE to address the class-imbalance problem while remodeling the behaviors of the environment agent for better performance. Experiments are conducted on NSL-KDD datasets. Comparative evaluations and their results are presented and analyzed. Using techniques such as SMOTE, ROS, NearMiss1 and NearMiss2, performance measures obtained from our simulations have led us to recognize specific performance trends. In particular, the proposed model AESMOTE outperforms AE-RL in several cases. Experiment results show an Accuracy greater than 0.82 and a F1 greater than 0.824.

3.PROPOSED SYSTEM

The Data Analysis stage examines the data and its parameters to determine if there are any data redundancies that could affect the prediction output. If a dataset has any parameters that aren't important, those values are eliminated. This phase also examines data to see whether it may be integrated in order to improve model prediction. Data Filtration phase, all empty/redundant values are removed from the data. The Train-Test Split Phase splits data into two halves, one for training and the other for testing. For example, data is split into two halves, with seventy percentage training data and thirty percentage test data. Phase of data-scaling The model gets data that is scaled to the model's specifications. This is always done by standardization or normalization. The objective of this work is to constrain the data to a certain range. This step reshapes data to make it more suited for the model in this method. We use the sklearn package of Python in the Model-Building Phase, which offers various packages for classification and regression tasks. We examine the accuracy of two commonly used classifiers XGBoost, Decision tree, and choose the best model by comparing their accuracy. It has been discovered that XGBoost outperforms the other models. Prediction Phase We test our model with the test input data and make a prediction

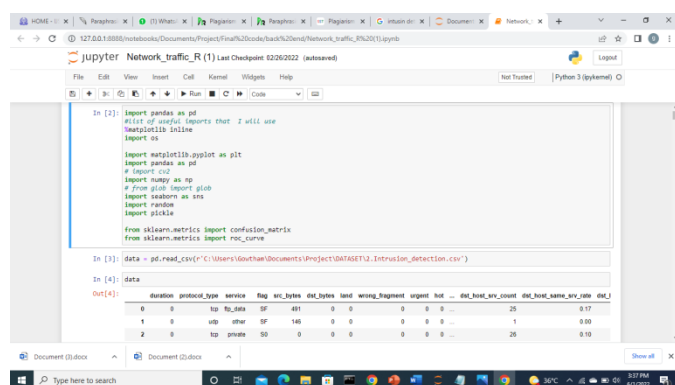
in this phase. The output is then compared against testing data in order to determine loss and accuracy. The XGBoost and decision tree accuracies are being tracked.

3.1MODULES:

- 1) Data Loading
- 2) Data Preprocessing
- 3) Data Cleaning
- 4) Data Splitting
- 5) Data Training
- 6) Model Testing

MODULE 1: DATA LOADING

The process of copying and loading data or data sets from a source file, folder, or programmer to a database or other application is known as data loading. Copying digital data from a source and pasting or loading the data into a data storage or processing application is the most common method. In database-based extraction and loading procedures, data loading is used. Such data is usually loaded into the destination application in a format that differs from the original source location. When data is copied from a word processing file to a database application, for example, the file format is converted from.doc or.txt to.CSV or DAT.



```

In [2]: import pandas as pd
        #list of useful imports that I will use
        import os
        import matplotlib.pyplot as plt
        import pandas as pd
        # import csv
        import numpy as np
        # from glob import glob
        import random as sns
        import pickle

        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import roc_curve

In [4]: data = pd.read_csv("C:\Users\iduchan\Documents\Project1\DATASET1\Intrusion_detection.csv")

Out[4]:
duration  protocol_type  service  flag  src_bytes  dst_bytes  land  wrong_fragment  urgent  host ...  dst_host_srv_count  dst_host_name  srv_rate  rate_i
0         0            top         sf         0         0         0         0         0         0         0 ...              25         0.17
1         1            top         sf         0         0         0         0         0         0         0 ...              1         0.00
2         0            top         sf         0         0         0         0         0         0         0 ...              26         0.10

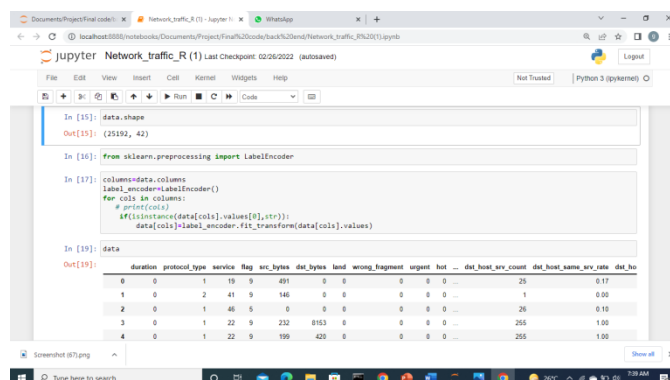
```

Figure 1: Dataset is loaded.

MODULE 2: DATA PREPROCESSING

Missing values were imputed in order to ensure that all algorithms could handle them. Conversely, certain algorithms, such as XGBoost, could automatically handle with incomplete data without imputation. The missing values were imputed based on their data type to keep the comparison simple. The median value of the entire elements is used to replace

missing values in numerical data formats. The missing entries in categorical data were replaced by the mode value of the complete entries.



```

In [15]: data.shape
Out[15]: (25192, 42)

In [16]: from sklearn.preprocessing import LabelEncoder

In [17]: columns=data.columns
        label_encoder=LabelEncoder()
        for cols in columns:
            # print(cols)
            if(isinstance(data[cols].values[0],str)):
                data[cols]=label_encoder.fit_transform(data[cols].values)

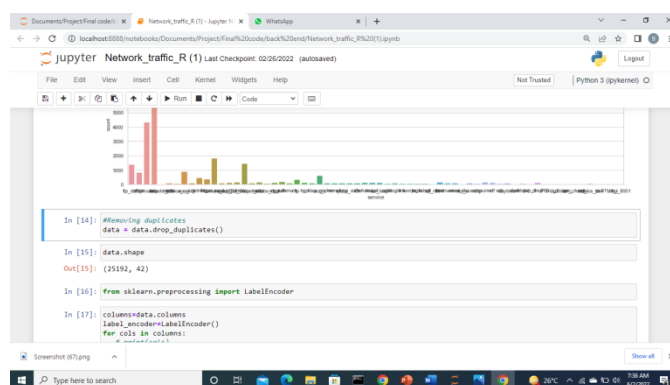
In [19]: data
Out[19]:
duration  protocol_type  service  flag  src_bytes  dst_bytes  land  wrong_fragment  urgent  host ...  dst_host_srv_count  dst_host_name  srv_rate  rate_i
0         0            top         sf         0         0         0         0         0         0         0 ...              25         0.17
1         1            top         sf         0         0         0         0         0         0         0 ...              1         0.00
2         0            top         sf         0         0         0         0         0         0         0 ...              26         0.10
3         0            top         sf         0         0         0         0         0         0         0 ...              255        1.00
4         0            top         sf         0         0         0         0         0         0         0 ...              255        1.00

```

Figure 2: Data is preprocessed.

MODULE 3: DATA CLEANING

The data is cleansed in this module. After the data has been cleaned, it is sorted according to the requirements. Data clustering is the term for this type of data grouping. Then look to see if there are any missing values in the data set. If a value is absent, replace it with any default value. After that, if any data has to be formatted, it can be done. Data pre-processing refers to the entire process that occurs before a prediction is made. Following that, the data is used to make predictions and forecasts.



```

In [14]: #Removing duplicates
        data = data.drop_duplicates()

In [15]: data.shape
Out[15]: (25192, 42)

In [16]: from sklearn.preprocessing import LabelEncoder

In [17]: columns=data.columns
        label_encoder=LabelEncoder()
        for cols in columns:

```

Figure 3: duplicated and null data is cleaned

MODULE 4: DATA SPLITTING

We divided the total dataset into 70 percent training and 30 percent test sets for each trial. The training set was utilized for resampling, hyper parameter adjustment, and training the model, whereas the test set was used to evaluate the trained model's performance. We specified a random seed (any random number) when splitting the data, ensuring that the data split was consistent every time the program ran.

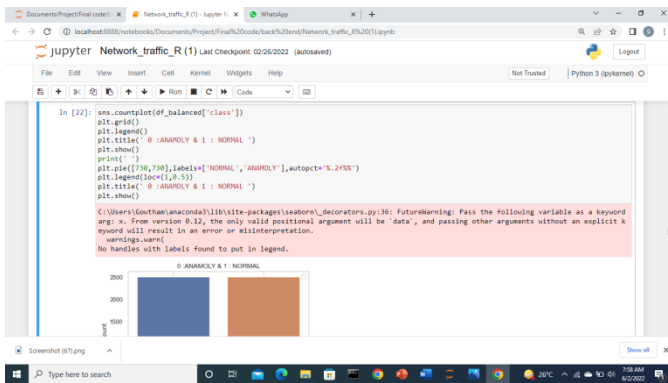


Figure 4: Data is spitted

MODULE 5: DATA TRAINING

Algorithms learns from dataset. They instead use the training data to form relationships, gain understanding, make judgments, and assess their confidence. The model works better when the training data is good. In fact, the quality and quantity of your training data are just as important as the algorithms themselves in determining the success of your data project. the data you want to use for training is typically enriched or labelled. It's also possible that you'll need more of it to power your algorithms. However, the data you've saved is probably not yet ready to be used to train your classifiers. Because if you want to build a good model, you'll need good training data.

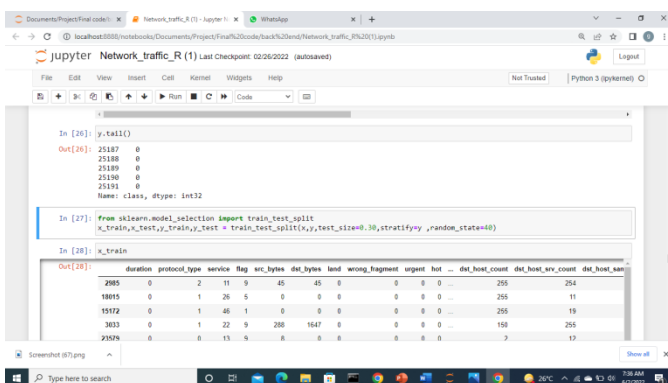


Figure 5: Data is trained for the model.

MODULE 6: MODEL TESTING

Hyper parameter is parameter which cannot gain knowledge from standard training method. These parameters express important aspect of this model that is how fast can it should learn. Applying machine learning algorithms.

DECISION TREE:

Decision tree is a type machine learning algorithm called supervised learning. The "forest" it

builds, is an ensemble of decision trees, usually trained with the "bagging" method. The bagging method is combines the learning models improve the overall result. Simply said, a decision tree combines numerous decision trees to produce a more accurate and stable prediction.

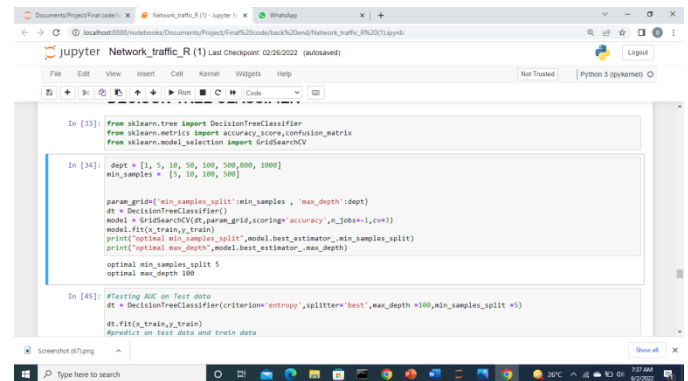


Figure 6: Decision tree algorithm.

XGBOOST ALGORITHM:

XGBoost algorithm is tremendously fast, and therefore it is a boosted decision tree, its speed and effectiveness. This classification prototype is utilized to improve the efficiency and speed of the model. XGBoost is a Machine Learning calculation that uses an inclination boosting approach that is based on decision trees.

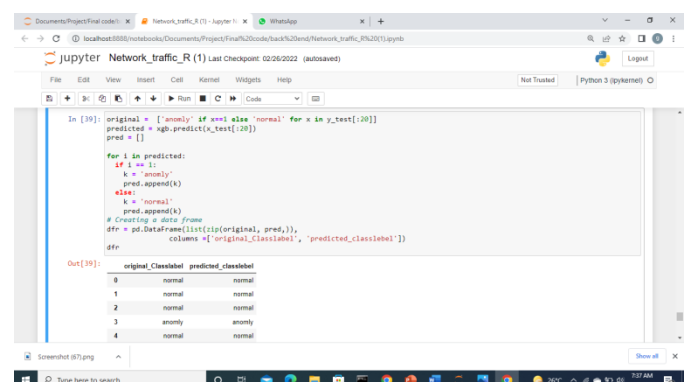


Figure 7: XGBoost algorithm

4.ARCHITECTURAL AND DATAFLOW DIAGRAM

4.1ARCHITECTURAL DIAGRAM:

The architectural diagram consists of Network dataset which the dataset undergoes process of data cleaning, null detection. Then the dataset is preprocessed and the dataset is now spitted 70 percentages for training and 30 percentages for test the model by using the decision tree and XGboost classifier the model is trained

and 30 percentage data is for the prediction to test the accuracy.

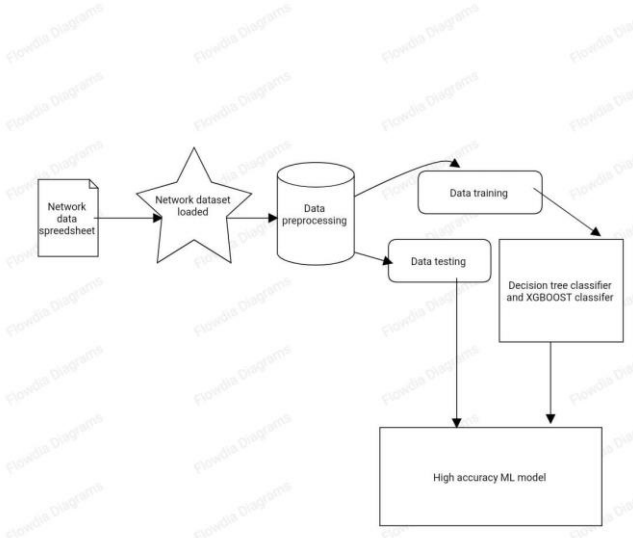


Figure 8: Architectural diagram.

4.2 DATAFLOW DIAGRAM:

Generally, data flows refer the flow of data and its functionality. In diagram describes that the collection of dataset and the dataset is cleansed and preprocessed. now the data is spitted for seventy percentage training data and thirty percentage testing data. After using the classifier like Decision tree and XGBoost the machine learning model is ready for the prediction of the data with high accuracy.

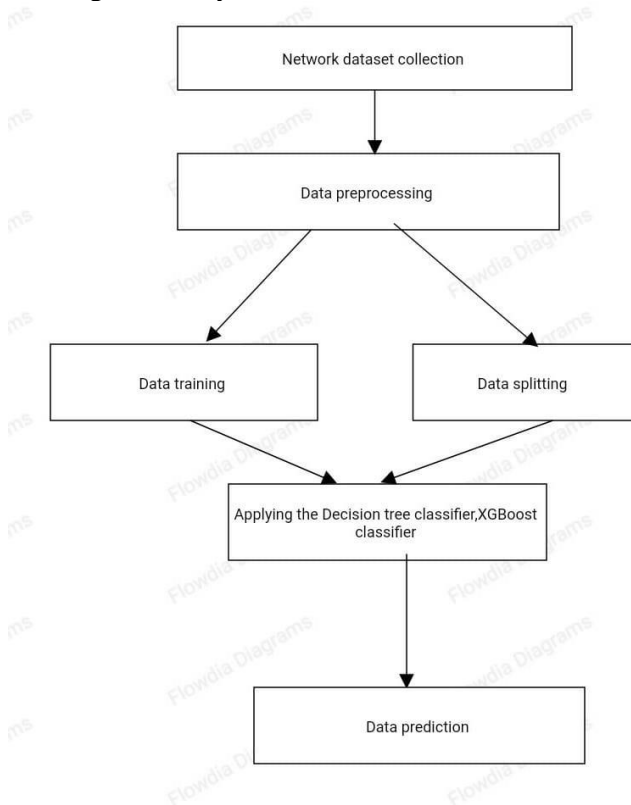


Figure 9: Dataflow diagram.

5. RESULTS AND OBSERVATIONS

The pre-processed data is used to make the prediction in this stage. This prediction can be made using any of the processes decision tree or XGBoost algorithm. The Linear Regression algorithm outperforms exceeds the other approach in prediction accuracy. As a result, the linear regression method is employed to make predictions in this project. The pre-processed data is separated for training and testing purposes. Then, using the test dataset value should be predicted should be same as trained dataset value. the dataset is then used to forecast data for the coming years.

TESTING:



Figure 10: Using Decision tree classifier Example.

PREDICTION:

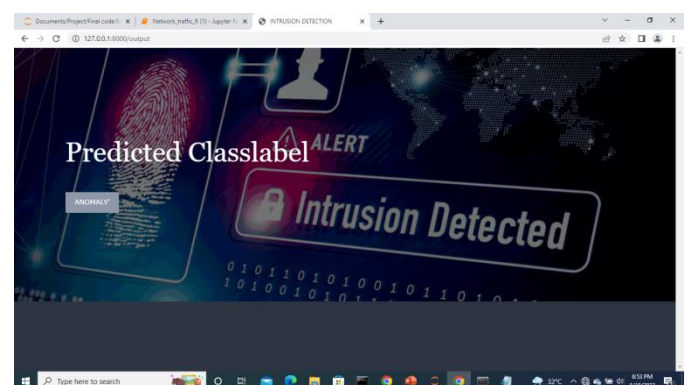


Figure 11: Data predicted Example.

TESTING:

[12] Stefan Seufert and Darragh O'Brien. 2017.

Machine Learning for Automatic Defence Against
Distributed Denial of Service Attacks.

[13] Mohammed Salem and Helen Armstrong. 2018.

Identifying DOS Attacks Using Data Pattern Analysis.